Side-Channel Attacks

Cryptographic algorithms assume that secret keys are utilized by implementations of the algorithm in a secure fashion, **with access only allowed through the I/Os**

Unfortunately, cryptographic implementations reveal information about internal operations in the power supply as they execute the algorithm

Power supply transient signals (I_{DDT}) and electromagnetic radiation can be measured using benchtop oscilloscopes and reverse-engineered to deduce the secret key

Dynamic current can measured by inserting a small resistance in series with the power supply wires, or by using current or EM probes

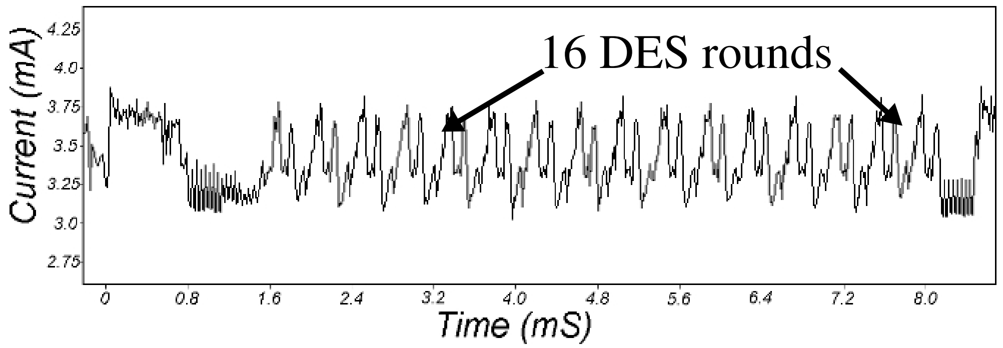
All types of hardware devices are vulnerable, including microprocessors, FPGAs and dedicated ASICs

Some are more vulnerable than others, e.g., microprocessors execute instructions on different functional units within the ALU

Therefore, I_{DDT} in these cases is dependent on both data and instructions

Simple Power Analysis (SPA)

SPA or Simple Power Analysis is a technique that involves directly analyzing I_{DDT} during cryptographic operations



P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", Proceedings of Advances in Cryptology-CRYPTO'99, Springer-Verlag, 1999, pp. 388-397

Above trace is measured over 1 ms, sampled at 5 MHz (5000 pts)

Blow-up of previous figure showing 2nd and 3rd rounds of DES encryption

28-bit DES key registers C and D
rotated once in round 2 and twice in round 3

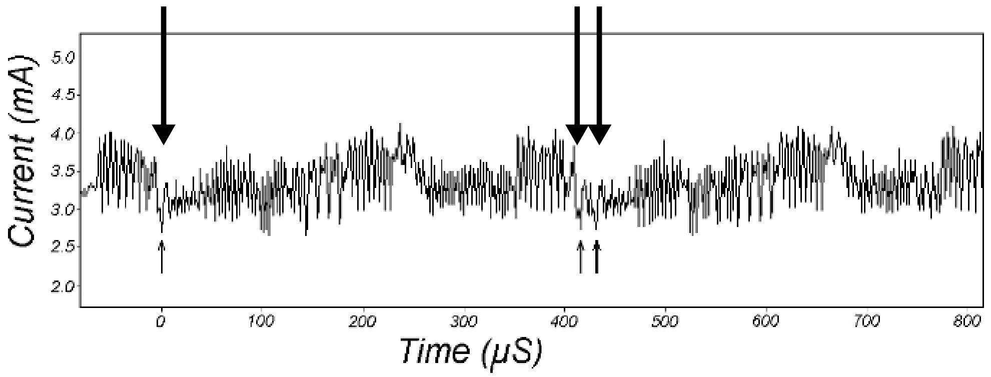


Figure 2: SPA trace showing DES rounds 2 and 3.

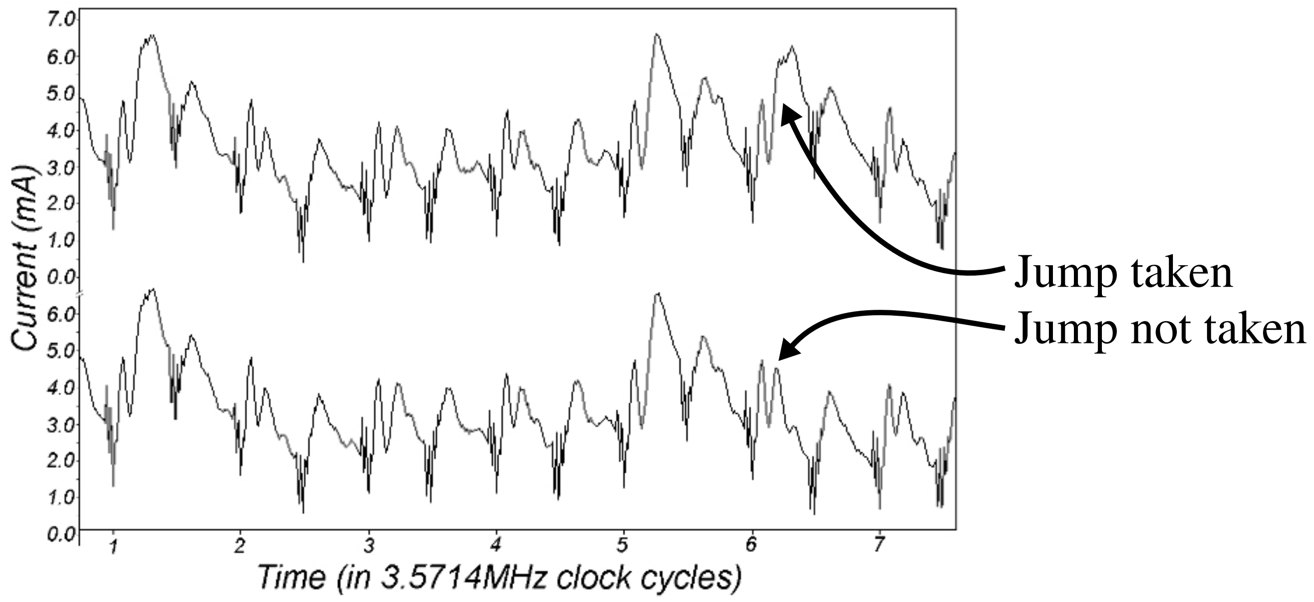
Small variations between the rounds are also visible, which can be exploited b/c they result from conditional jumps based on key bits, etc.

Simple Power Analysis (SPA)

The DES key schedule algorithm

```
Algorithm $KeySchedule(K)$    // $|K| = 56$
    $K \leftarrow PC\text{-}1(K)$
    Parse $K$ as $C_0 \| D_0$
    for $r = 1, \ldots, 16$ do
        if $r \in \{1, 2, 9, 16\}$ then $j \leftarrow 1$ else $j \leftarrow 2$ fi
        $C_r \leftarrow leftshift_j(C_{r-1})$ ; $D_r \leftarrow leftshift_j(D_{r-1})$
        $K_r \leftarrow PC\text{-}2(C_r \| D_r)$
    return$(K_1, \ldots, K_{16})$
```

An even higher resolution trace through two regions, each with 7 clock cycles at 3.5714 MHz



Simple Power Analysis (SPA)

The differences in the traces result primarily from differences in the power consumption of **different microprocessor instructions**

It is clear that SPA can reveal the sequence of instructions executed

Therefore, it can be used to break crypto. implementations in which the execution path depends on the data being processed

For example:

- **DES Key Schedule**

  The key schedule involves rotating 28-bit key registers

  A conditional branch is commonly used to check the bit shifted off the end so that "1" bits can be wrapped around

  The resulting power consumption traces for a "1" bit and a "0" bit will contain different SPA features if the execution paths takes different branches for each

Simple Power Analysis (SPA)

- **DES permutations**

  DES implementations perform a variety of **bit permutations**

  Conditional branching in software or microcode can cause significant power consumption differences for "0" and "1" bits

- **Comparisons**

  String or memory comparison operations typically perform a conditional branch when a mismatch is found

  This conditional branching causes large SPA (and sometimes timing) characteristics

- **Multipliers**

  Modular multiplication circuits tend to leak a great deal of information about the data they process

  The leakage functions depend on the multiplier design, but are often strongly correlated to operand values and Hamming weights

Simple Power Analysis (SPA)

- **Exponentiators**

  A simple modular exponentiation function parses the exponent, performing a squaring operation in every iteration

    An additional multiplication occurs for each exponent bit that is equal to "1"

  The exponent can be compromised if squaring and multiplication operations have different power consumption characteristics

    Or take different amounts of time

Preventing SPA

Avoiding procedures that use secret intermediates or keys for conditional branching operations will mask many SPA characteristics

The microcode in some microprocessors cause large operand-dependent power consumption features, which are vulnerabilities

ASIC implementations of symmetric cryptographic algorithms have sufficiently small power consumption variations, so SPA cannot be used to extract keys

Differential Power Analysis (DPA)

In addition to large-scale power variations due to the instruction sequence, there are effects correlated to **data values** being manipulated

These variations tend *to be smaller* and are sometimes overshadowed by measurement errors and other noise

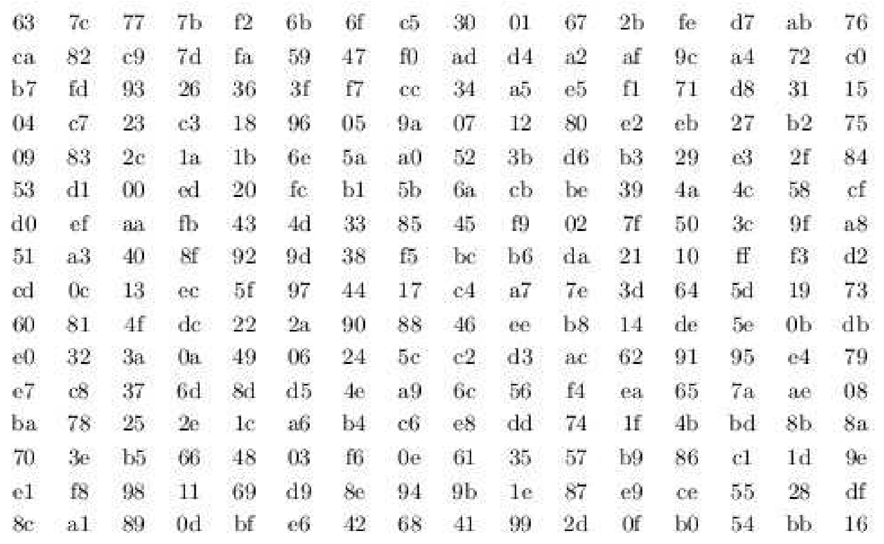
In such cases, it is still often possible to break the system using **statistical functions**

AES will be used here to demonstrate this

```
function AES$_K$(M)
    (K$_0$, ..., K$_{10}$) <- expand(K)
    s <- M XOR K$_0$
    for r = 1 to 10 do
        s <- S(s)
        s <- shift-rows(s)
        if r <= 9 then s <- mix-cols(s) fi
        s <- s XOR K$_r$
    endfor
    return s
```

Differential Power Analysis (DPA)

AES first XORs the subkey K_0 (which is in fact **the key** -- see key sched.) with the plaintext byte and then uses the result to ’lookup’ a value in the SBOX function



63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

This table lists the values of *S(0)*, *S(1)*, .. *S(255)*

Figure 2.9: The AES S-box, which is a function $S : \{0,1\}^8 \to \{0,1\}^8$ specified by the following list. All values in hexadecimal. The meaning is: $S(00) = 63$, $S(01) = 7c, \ldots, S(ff) = 16$.

Therefore, each output byte in the first round can be computed given a specified, i.e., known, plaintext byte and a *guess* of the subkey K_0 as:

S(*p* XOR *k*), with *p* representing the plaintext and *k* the subkey

Let’s focus on the high order bit, *v*, of the output byte

Differential Power Analysis (DPA)

The attack assumes the power consumption during processing of this high order bit will depend on whether it is a ’0’ or a ’1’

The attack begins by measuring the power supply curve as each of 1000 random (but known) plaintexts are encrypted

Once these traces are obtained, we have ALL the information we need to learn the key, i.e., the rest of the attack focuses only on post-processing these wfms

The process of ’learning’ *k* simply involves creating two subsets of the 1000 power trace wfms for each *guess* of *k*

- Given that *k* can be 1 of 256 different values, this requires this process to be repeated 256 times, once for each guess

The two subsets are created by computing the value of *v* under each guess of *k*

- Remember, the plaintext bytes are known

- Therefore, for each guess of *k*, each is XORed and run through SBOX

Differential Power Analysis (DPA)

The bit value that is generated for *v* is used to define the subsets, with all plaintext wfms that generate a ’0’ for *v* in one subset, and those generating a ’1’ in the other

An important part of the attack is to eliminate the ’background noise’

Background noise is generated by **simultaneous switching activity** on other SBOX output bits during the *time region of interest*

The *time region of interest* is simply the period of time when AES is computing the XOR and subsequently carrying out the SBOX operation

Background noise is eliminated by computing an **average trace** using all wfms in each of the subsets

The assumption here is that each of the other 127 bits on the SBOX outputs in each wfm subset have nearly equal numbers 0’s and 1’s

If this is true, then the average power trace will become **flat** for these bits because of the averaging, effectively cancelling them out

Differential Power Analysis (DPA)

The final step is to compute a **difference trace** from the two averages, for each guess of the subkey *k*

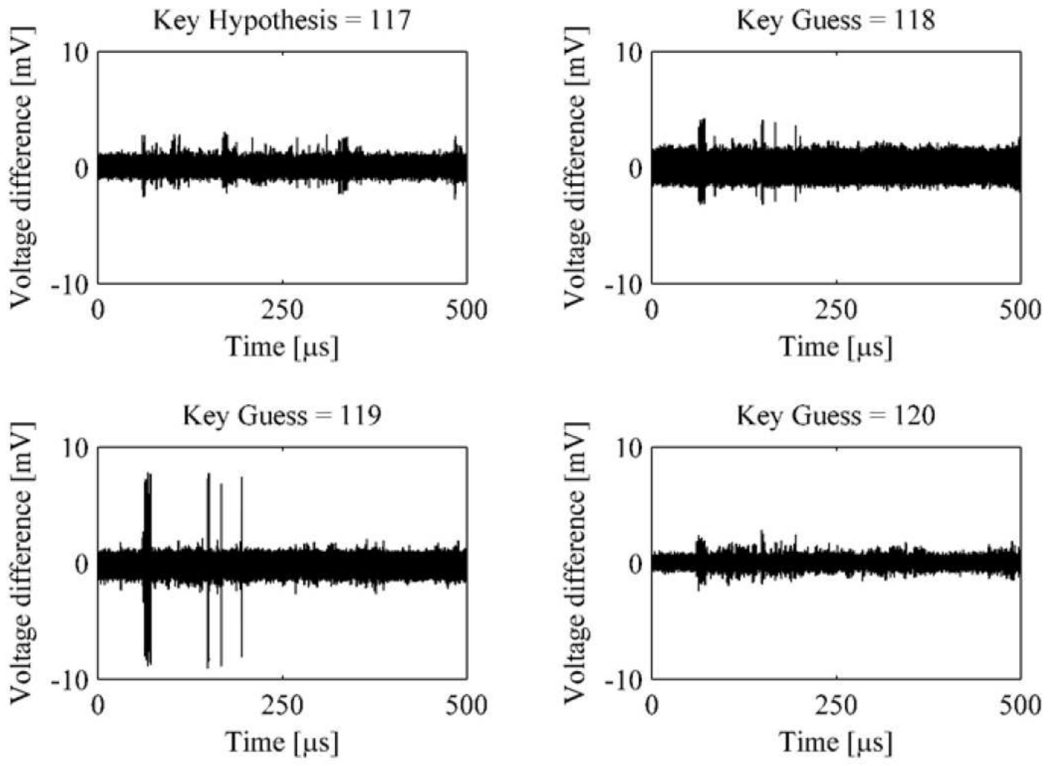


Figure 1.5. Difference plots for the key guesses 117, 118, 119, and 120.

S. Mangard, E. Oswald and T. Popp, " Power Analysis Attacks: Revealing the Secrets of Smart Cards", Springer, 2007

The above shows guesses for *k* of 117, 118, 119 and 120

It is clear from the spikes that the actual subkey *k* used by AES is 119

Differential Power Analysis (DPA)

DPA works because it can inspect *intermediate* values in the computation process, something the authors of AES assumed was not possible

Power analysis provides internal observation of the temporal and spatial switching behavior of the chip

This feature has been heavily used in the manufacturing test community since the early 90's!

So DPA was not first to make use of power supply transient signal analysis

My Ph.D. research in 1997 (see ITC'96 and '97) is also based on an analysis of I_{DDT} signals

However, unlike manufacturing test methods, such as I_{DDQ} and I_{DDT}, DPA does NOT need to account for process variation effects between ICs

This is true because DPA is carried out entirely on ONE chip

This allows much higher levels of resolution to be achieved